DATA ERROR CONTROLS

IN

INFORMATION SYSTEMS DESIGN

DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY
KANPUR

DATA ERROR CONTROLS

IN

INFORMATION SYSTEMS DESIGN

By

P S KENJALE

A thesis submitted in partial fulfilment

of the requirements for the degree

of

MASTER OF TECHNOLOGY

To

THE DEPARTMENT OF ELECTRICAL ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR (INDIA)

MAY 1974.

CERTIFICATE

Certified that this thesis entitled "Data Error Controls in Information Systems Design" was carried out under my supervision by Mr P S Kenjale and has not been submitted elsewhere for a degree

(V RAJARAMAN)
Professor
Electrical Engineering Department
I I T ,Kanpur

ACKNOWLEDGEMENTS

Professor V Rajaraman created in me a love for research He guided me right through my technical and non-technical difficulties, and kept me ever on the right track I can never forget the "eureka" moments I shared with him He gave me the methodology of research and the confidence required to tackle any big problem

As a gesture of my gratitude, I accept Professor V Rajaraman as my lifelong GURU

I thank Sri S S Pethkar for his excellent and pains-taking job of typing, as can be seen in the thesis

- P S Kenjale

ABSTRACT

The following work deals with the control of data errors in information systems After a brief survey, the inadequacy of the existing methods is established It is submitted that in many application areas, automatic error correction is essential A novel single and exchange error correcting code is developed A quantitative method, based on a model, is given, to choose from amongst the data entry methods

Next, the problem of secrecy transformations in system design is considered Some simple algorithms are presented, a method of complicating them is indicated, together with a quantitative method of evaluating them

Finally, a live case is taken up to illustrate the above concepts

C O N T E N T S

CHAPTER 1

INTRODUCTION

A characteristic invariably found in all information systems is that the volume of data is large and the processing required on the date is simple Usually about 80% of the work done on the computer is for input editing, procedure control, output editing and formatting A large part of this is to reduce the effect of data errors on processing

Inspite of its great importance, this problem has not received adequate attention, and as a result, input data editing is, at present, very simple and largely inadequate Usually several checks are made in every system, but each of the checks is only a necessary condition for the correctness of data, and none of them is a sufficient condition Consequently, inspite of the most elaborate editing procedures, a number of errors still persist in the data input to the processor All the errors which are detected have to be corrected manually, a process which involves a large amount of delay Also, if a batch of data is large, then there could be errors in the correcting of errors detected previously, and hence, the problem becomes more severe

The present work mainly deals with the above problem In Chapter 2, the existing data error control methods are discussed The

main method is editing This includes error detecting which is possible by proper coding, some properties of individual fields, and interfield relationships Batch controls and cross-footing checks also permit some error detection But the most interesting method is by the inclusion of check digits After analysing such methods, the need for error-correcting is brought out

Chapter 3 presents a proposed error correcting code The method is described, proofs are given, followed by some comments on the algorithm

Chapter 4 compares the data entry method using this error-correcting code, with some of the well-known existing methods The comparision is effected by means of a simple model developed for this purpose, and results are presented

Chapter 5 considers secrecy transformations, a topic not quite related with data error controls Some of the data in data bases is secret and has to be protected In some applications secret data has to be handled manually, and it becomes necessary to code it This problem is analysed, some simple algorithms are given, and a quantitativemethod of evaluating the algorithms is developed

Chapter 6 takes up a case study which, apart from being live, is well suited for the application of the concepts developed herein

Chapter 7 is the conclusion Apart from discussing what has been achieved, it also recommends the application areas in which the methods developed can be profitably employed

A comprehensive bibliography is given for the interested research workers

CHAPTER 2

EXISTING DATA ERROR CONTROL METHODS

2 1 Overview :-

This chapter briefly reviews the existing data error control methods All of them are in the form of checks Some of the checks are natural, and others are artificially introduced Examples of natural checks are the validity checks possible in significant digit codes, limit checks on fields, inter-field relationships, etc Examples of artificial checks are record count, control total, cross-footing checks, etc The main contention is that these checks are too naive and elementary The only interesting check is the modulus 11 check digit Though these are useful to point out certain errors, it is pointed out that they are in no way complete, and errors can still persist Scientific investigation is required here In particular, a simple error correcting code is required which is acceptable for information systems

2 2 Editing :-

Editing is a procedure in which all possible checks are made on input data, to point out as many errors as possible The checks used at present are

1) Record count
2) Control totals
3) Proof figures
4) Type checks
5) Limit checks
6) Inter field checks
7) Cross-footing checks
8) Tape and disk labels
9) Check point and restart
10) Sequence check

Record count is the number of records in a batch It is found manually, and put as a special record after the batch The number of records read are counted and the total is tallied with this record count The aim is to discover whether there are omissions or duplications of records Control totals can be used for numeric fields The same field in all the records of a batch are added to get a control total, whether the total makes sense or not While reading, the computer also finds this total and tallies it with the total fed in Proof figure is a number carried with a field which is a numeric value, such that their sum adds up to a constant decided upon previously Type checks involve checking the characters of a field which should be either purely numeric, or purely alphabetic Limit checks can be performed on numeric

fields Usually the quantity in any field can lie only in some range, and this check sees whether it actually does Interfield checks can point out and even correct some checks For instance if identification codes lie in some ranges for group A, and some other ranges for group B (whatever that means), and if a batch of transaction records is fed in which all identification codes are present in increasing order, then if it is found that the group is mentioned as B and the code lies in the A-range, then the group can be safely corrected Cross-footing checks essentially involve doing an operation in two ways and tallying the result Tape and disk labels are useful to check whether the correct batch is being brought in, and they also contain some other information Check point and restart is actually not a part of editing, they are points in a program where the processing is proved and enough information is stored to restart processing at that point

2 3 Error detecting codes :-

The modulo 11 check digit scheme is the only error detecting code worth mentioning But before describing it, the various errors possible will be mentioned Let a number be

$$N = n_1 n_2 \quad n_i \quad n_k$$

where each n_i can be 0, 1, 2,   , 9 (Extension to alphanumeric codes is simple), The "channels' which can produce errors are reading and

copying, reading and punching, etc The errors are

1 Single transcription error are n_i becomes n_i'

2 Single transposition error n_i and n_j become n_j and n_i

3 Multiple identical adjacent digit transcription error $n_i, n_{i+1} \quad n_j$ which are all equal to x become y y y

4 Zero shift errors The number of adjacent zeroes gets changed

In addition, there are several kinds of multiple errors

Modulus N check digits are of two types In the first type, a check digit n_{k+1} is added such that

$$\sum_{1}^{k+1} n_i w_i \equiv 0 \text{ (modulo N)}$$

where w_i is a weight associated with the ith digit In the second type, the weights are $1, 10, 10^2$, etc , so that the number itself, when divided by N gives the check digit Systems of the second type can be reduced to the first type, so that it is enough to consider only these

The problem is to find N and the weights w_i so that most of the errors can be detected

Single transcription errors :-

Here n_i becomes n_i' So $\left[\sum_{1}^{k+1} n_i w_i\right]_N = S$ which should be zero, becomes $(n_i' - n_i)w_i$ The checking involves seeing if S is zero So this error can go undetected if

$$(n_i' - n_i)w_i \;=\; l\,N, \quad l = 1,2,3,$$

This can be avoided if $n_i < N$, $w_i \leqslant N$ and N is prime

<u>Single transposition errors</u> -

Here n_i and n_j get interchanged So S becomes $(n_i - n_j)(w_j - w_i)$, which can go undetected if

$$(n_i - n_j)(w_j - w_i) = l\;N, \qquad l = 0,1,2,3,$$

and it can be avoided if $n_i \leqslant N$, $w_i < N$, no two weights are equal, and N is prime

<u>Identical adjacent digit transcription errors</u> -

Here n_i, n_{i+1} $\quad$ n_j which are all equal to x become yy $\quad$ y, so that S becomes

$$(y - x)\,(w_i + w_{i+1} + \quad + w_j)$$

This can go undetected if

$$(y-x) \sum_i^j w_k = l\,N, \qquad l = 1,2,3,$$

So to avoid this, it is enough if $n_i \leqslant N$, $\sum_i^j w_k \neq l\,N$, $l = 1,2,3,$

for $j > i$, and N is prime

So these errors can be detected 100 % But for other errors, simple analysis is not possible However, assuming equal probability, $(N-1)/N$ of them can be detected

D F Beckley [4] in 1967 gave the following weights, with $N = 11$

$w_1 = 1$, $w_2 = 2$, $w_3 = 5$, $w_4 = 3$, $w_5 = 6$, $w_6 = 4$, $w_7 = 8$, $w_8 = 7$, $w_9 = 10$, $w_{10} = 9$

In 1969, W G Wild [46] developed the theory as given above In addition to $N = 11$, another suggestion was $N = 97$ which requires two check digits

The modulus 11 check digit which enables $\left[\sum_1^{k+1} n_i w_i\right]_{11} = 0$ is

$$n_{k+1} = \left[11 - \left[\sum_1^k n_i \ w_i\right]_{11}\right]_{11}$$

Now it is possible that $n_{k+1} = 10$, in which can it cannot be accomodated in one decimal digit

One way to overcome this is to discard all members which give a check digit of 10 This can be done for most of the systems in the design phase, but not for systems already designed Another way is to use some special symbol like A for 10 But this creates problems in editing D V A Campbell [11] came up with a solution to this in 1970

He observed that Beckley's weight sequence is only one of the possible ones He gave another sequence W', where $w_1' = 1$, $w_2' = 9$, $w_3' = 6$, $w_4' = 8$, $w_5' = 7$, $w_6' = 5$, $w_7' = 4$, $w_8' = 4$, $w_9' = 3$ and $w_{10}' = 10$ According to him if a number gave a check digit of 10 with the weights W, it would not, with weights W' So whenever a number gave a check digit of 10, the W' system was to be used In the checking algorithm, if $S = 1$, then S was to be recalculated using W' R W Broderick and C J Reid [36] pointed out some defects in this systems For instance, 1003 and 180000000 give a check digit of 10 in both the systems Reid proved that it is sufficient to have $w_i' \equiv w_i$ (modulo 11) for all i except the weight applicable to the check digit and its arithmetic inverse modulo 11, for which, $w_i' = w_i$

The next important paper was by A M Andrew in 1970, who gave a variant of the modulus 11 scheme, in which a check digit of 10 cannot occur Instead of evaluating $\left[\sum_1^k n_i w_i\right]_{11}$, he proposed

$$\left[\sum_1^k \left[(n_i + a(w_i)) \quad w_i\right]_{11}\right]_{10}$$

But this cannot detect all the errors the original system can

T Briggs[8] in 1970, analysed all the work done so far, and brought out some practical aspects. He found a tree-diagram method of generating the Wild weight sequence, and modified Campbell's method, removing Reid's objection

Extension of the modulus 11 scheme to alphanumeric data is straightforward It is also possible to use the character code of any computer instead of assigning sequential numbers to the characters

2 4 The need for error correcting :

The check digit system described above only detects most of the errors in input data Corrections have to be made manually, and to that extent it is just another edit procedure This is time consuming and if the data is very voluminous, it could involve going several times "around the loop ' This does not enable maintaining an up-to-date file, and consequently results in bad management information If the error rate is high, the effect could be disastrous and credibility on the computer may be lost,

What is needed is a simple way of automatically correcting most, if not all, of the errors This would make the system very efficient But methods similar to those in coding theory cannot be used, since they would involve a tremendous overhead of check digits, and consequently will not be accepted A little extra computer time can, however, be tolerated

The next chapter proposes one error correcting code meeting the above requirements

APPENDIX

A simple manual method of calculating check digits

For large amounts of data generated manually for which it is feasible to add check digits, it is cumbersome and error-prone to calculate them for each number using the formula

$$\text{Check digit} = \left[N - \left[\sum_{i=1}^{k} n_i w_i \right]_N \right]_N$$

A practical method is presented here for N = 11 It holds for any N It uses the result

$$\left[\sum_{i=1}^{k} n_i w_i \right]_{11} = \left[\sum_{i=1}^{k} \left[n_i w_i \right]_{11} \right]_{11}$$

It consists of two tables Table 1 gives the values of $\left[n_i w_i \right]_{11}$ for any i They are to be added manually The sum, when referred to Table 2, gives the check digit

Beckley's weights have be chosen here :

w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8	w_9	w_{10}
1	2	5	3	6	4	8	7	10	9

Let $n_1, n_2 \quad n_k$ be the number, and it is required to find n_{k+1}, the check digit such that $\left[\sum_{i}^{k+1} n_i w_i\right]_{11} = 0$, with the weights as given, and $w_{k+1} = 1$

Algorithm -

1 Look up Table 1 Under the digit 1 column, look up for the number corresponding to n_1 Similarly look up the number for the other digits and add them up to give sum

2 Look up Table 2, locate the sum and read the corresponding check digit

If it is required to check manually whether a number is correct, the same procedure can be applied, with the difference that instead of going to Table 2, it is enough to check if the sum is 0, 11, 22, 33, etc

TABLE 1

Weight	9	10	7	8	4	6	3	5	2	1
Actual Digit	Digit 1	Digit 2	Digit 3	Digit 4	Digit 5	Digit 6	Digit 7	Digit 8	Digit 9	Digit 10
0	0	0	0	0	0	0	0	0	0	C
1	9	10	7	8	4	6	3	5	2	1
2	7	9	3	5	8	1	6	10	4	2
3	5	8	10	2	1	7	9	4	6	3
4	3	7	6	10	5	2	1	9	8	4
5	1	6	2	7	9	8	4	3	10	5
6	10	5	9	4	2	3	7	8	1	6
7	8	4	5	1	6	9	10	2	3	7
8	6	3	1	9	10	4	2	7	5	8
9	4	2	8	6	3	10	5	1	7	9

Legend -

If the digit in the ith place is n_i, this table gives the value of $\left[n_i w_i\right]_{11}$ To locate this value, go horizontally to locate digit number, and vertically to match the actual digit The 10th digit is the check digit, to be used only when checking the correctness of a given number

TABLE 2

Sum										Check digit
0	11	22	33	44	55	66	77	88	99	0
1	12	23	34	45	56	67	78	89	100	10
2	13	24	35	46	57	68	79	90	101	9
3	14	25	36	47	58	69	80	91	102	8
4	15	26	37	48	59	70	81	92	103	7
5	16	27	38	49	60	71	82	93	104	6
6	17	28	39	50	61	72	83	94	105	5
7	18	29	40	51	62	73	84	95	106	4
8	19	30	41	52	63	74	85	96	107	3
9	20	31	42	53	64	75	86	97	108	2
10	21	32	43	54	65	76	87	98	109	1

Legend - For any sum which the number to be coded, produces, from Table 1, this table gives directly the check digit

CHAPTER 3

A PROPOSED ERROR CORRECTING CODE

3 1 Overview :-

The need for automatic error correction was established in the previous chapter G M Weinberg[4] in 1961, extended Hamming's single error correcting code to decimal members It was a straightforward extension using modulo 10 arithmetic instead of the modulo 2 arithmetic of Hamming It could correct all single errors, but it required k check digits for m information digits, such that $2^k \geqslant m + k + 1$ For example, for 4 information digits, 3 check digits were required The method was not free of pitfalls After this, no further work was done in this area

The proposed error correcting code requires only two extra check digits irrespective of the number of information digits, and it corrects all single errors (i e , single transcription errors) and all single transposition errors, which together account for about 95% of all the errors First the method is presented Then the proofs are given and the theory underlining the method is developed Finally, the algorithm is discussed

3 2 The method :-

Let the number to be coded be $N = n_1, n_2, \quad n_i \quad n_k$ where $n_i \in \{0, 1, 2, \quad , 9\}$ for $i = 1, 2, \quad , k$

Procedure for coding :-

Let a weight sequence

$$W = (w_1\ w_2 \qquad w_{k+2})$$

be, for $k = 8$,

4 6 10 9 3 7 8 5 2 1

and another sequence

$$W' = (w_1'\ \ w_2' \qquad w_{k+2}')$$

be 5 10 4 8 9 6 1 7 3 2

Two check digits n_{k+1} and n_{k+2} are to be appended to N, given by

$$n_{k+1} = \left[2\left(-\sum_1^k n_i w_i\right) + \sum_1^k n_i w_i'\right]_{11}$$

$$n_{k+2} = \left[2\left(-\sum_1^k n_i w_i'\right) + 3\left(\sum_1^k n_i w_i\right)\right]_{11}$$

The coded number is

$$N' = n_1\ n_2 \qquad n_{k+1}\ \ n_{k+2}$$

Procedure for error detecting :-

Check whether

$$S_1 = \left[\sum_1^{k+2} n_i w_i\right]_{11} = 0$$

If yes, the number is error free Otherwise, call the procedure for error correcting

Procedure for error correcting :-

1 Compute

$$S_2 = \left[\sum_{1}^{k+2} n_i w_i'\right]_{11}$$

2 Consider each digit from 1 to k+2 For each, insert all the other 9 possible digits At every insersion, compute S_1 If $S_1 = 0$ at some insersion, compute S_2 If $S_2 = 0$, then the error was a single transcription error which has been corrected Exit Else continue, replacing the original digit in its place after all the other 9 digits have been inserted unsuccessfully

3 Exchange the first and second digits Compute S_1 If $S_1 = 0$, compute S_2 If $S_2 = 0$, it was a single transposition error which has been corrected Exit Otherwise put back the digits Next exchange the second and third digits, and so on

4 Repeat (3), but now exchange alternate digits

5 Report "an error which cannot be corrected" Exit

4 3 Proofs:-

The weight sequences

$W = w_1 \; w_2 \qquad w_{k+2}$

$W' = w_1' w_2' \qquad w_{k+2}'$

are chosen so as to individually satisfy the requirements of the modulus 11 error detecting scheme explained in the previous chapter So they detect the same errors Additional constraints will now be placed on W and W to effect correction

Single transcription errors -

Now

$$S_1 = \left[\sum_{1}^{k+2} n_i w_i\right]_{11},$$

$$S_2 = \left[\sum_{1}^{k+2} n_i w_i'\right]_{11}$$

If there is no error, $S_1 = 0$ and $S_2 = 0$ Due to a single transcription error, if the digit n_i becomes n_i', then

$$S_1 = (n_i' - n_i)w_i,$$

$$S_2 = (n_i' - n_i)w_i'$$

It is enough to prove that there is a unique way of correcting this error, that is, a unique way of making both S_1 and S_2 zero Obviously there is one way, namely by changing n_i' to n_i To prove that there is no other way, consider any other change, say n_j to n_j' This results in the addition of

$$S_1' = (n_j' - n_j)w_j$$
$$S_2' = (n_j' - n_j)w_j'$$

to S_1 and S_2 If this should not make S_1 and S_2 both zero, then it is sufficient if

$$\underline{\text{not}} \left[S_1 = -S_1' \quad \underline{\text{and}} \quad S_2 = -S_2' \right]$$

That is,

$$(n_i' - n_i)w_i = -(n_j' - n_j)w_j$$

and

$$(n_i' - n_i)w_i' = -(n_j' - n_j)w_j'$$

should not hold simultaneously If one of them holds, the other should not So dividing one by the other,

$$\frac{w_i'}{w_i} = \frac{w_j'}{w_j}, \quad i,j = 1,2, \quad , k+2$$
$$i \neq j$$

should not hold. This is therefore a sufficient condition for the unique correctability of all single transcription errors.

<u>Single transposition errors:-</u>

If due to this error, two digits n_i and n_j get interchanged, then

$$S_1 = (n_i - n_j)(w_j - w_i),$$
$$S_2 = (n_i - n_j)(w_j' - w_i')$$

One way of correcting this is to interchange again n_i and n_j To show that there is no other way to make S_1 and S_2 both zero, consider any other interchange, say of digits n_p and n_q This adds

$$S_1' = (n_p - n_q)(w_q - w_p),$$

$$S_2' = (n_p - n_q)(w_q' - w_p')$$

to S_1 and S_2 For unique correctability, as before, the condition is

$$\underline{\text{not}} \left[S_1 = -S_1' \quad \underline{\text{and}} \quad S_2 = -S_2' \right]$$

That is

$$(n_i - n_j)(w_j - w_i) = -(n_p - n_q)(w_q - w_p)$$

and

$$(n_i - n_j)(w_j' - w_i') = -(n_p - n_q)(w_q' - w_p')$$

should not hold simultaneously Hence, as before, a sufficient condition is

$$\frac{w_i' - w_j'}{w_i - w_j} \neq \frac{w_p' - w_q'}{w_p - w_q}, \qquad i, j, p, q = 1, 2, \quad , k+2$$
$$\text{and } i \neq j \text{ and } p \neq q$$

But usually single transposition errors are for adjacent and alternate digits Hence it is sufficient if

$$\frac{w_i' - w_{i+1}'}{w_i - w_{i+1}} \neq \frac{w_j' - w_{j+1}'}{w_j - w_{j+1}} ,$$

$$\frac{w_i' - w_{i+2}'}{w_i - w_{i+2}} \neq \frac{w_j' - w_{j+2}'}{w_j - w_{j+2}} ,$$

$$i, j = 1,2, \quad , k+2 \quad \text{and} \quad i \neq j$$

It remains to be shown that weight sequences W and W' exist as required above Instead of proving, W and W' have been found by trial and error That they satisfy the required conditions can be seen from the following table

i	10	9	8	7	6	5	4	3	2	1
W	4	6	10	9	3	7	8	5	2	1
W'	5	10	4	8	9	6	1	7	3	2
$\frac{w_i}{w_i'}$	$\frac{4}{5}$	$\frac{3}{5}$	$\frac{5}{2}$	$\frac{9}{8}$	$\frac{7}{3}$	$\frac{7}{6}$	8	$\frac{5}{7}$	$\frac{2}{3}$	$\frac{1}{2}$
$\frac{w_{i+1} - w_i}{w_{i+1}' - w_i'}$	$\frac{2}{5}$	$\frac{-3}{5}$	$\frac{-1}{4}$	-6	$\frac{-4}{3}$	$\frac{-1}{5}$	$\frac{-1}{2}$	$\frac{3}{4}$	1	-
$\frac{w_{i+2} - w_i}{w_{i+2}' - w_i'}$	-6	$\frac{-3}{2}$	$\frac{-7}{5}$	-1	$\frac{-5}{8}$	-2	-3	$\frac{4}{5}$	-	-

Calculation of check digits:-

The check digits n_{k+1} and n_{k+2} have to be found satisfying

$$\sum_{1}^{k+2} n_i w_i \equiv 0 \ (11)$$

$$\sum_{1}^{k+2} n_i w_i' \equiv 0 (11)$$

The existence of a unique solution is to be proved and the solution is to be found This is possible by using number theory, especially the work of Euler on Diophantine equations[23]

The following theorems and definitions are necessary:

Theorem 1 - Modulo any integer m,

$$a \equiv b \iff b \equiv a \iff b - a \equiv 0 \iff a - b \equiv 0$$

Definition 1 - If $x \equiv b(m)$, then b is a residue of x modulo m If $0 \leq b < m$, then b is a least positive residue

Definition 2:- A set of positive integers is a complete set of residues modulo m, if no two of them are congruent, and every integer is congruent to one of them

The set $\{0, 1, \quad , m-1\}$ is a complete set of least positive residues modulo m

Theorem 2:- Modulo any integer m,

1) $a \equiv b \Rightarrow ca \equiv cb$

2) $a \equiv b,\ c \equiv d \Rightarrow a + c \equiv b + d$

$$\Rightarrow ar + cs \equiv br + ds$$

$$\Rightarrow ac \equiv bd$$

Definition 3 - (m, c) is the g c d of m and c

Theorem 3 -

$$ca \equiv cb(m) \Rightarrow a \equiv b(m/(m, c))$$

Corollary 1:-

$$ca \equiv cb(m),\ (c,m) = 1 \Rightarrow a \equiv b(m)$$

Definition 4 - A residue class $A = \left\{ a \mid a \equiv r(m) \right\}$ is a prime residue class if $(r,m) = 1$

Definition 5 - A complete set of prime residues is a set $S = \left\{ r_i \right\}$ such that

1) $i \neq j \Rightarrow r_i \not\equiv r_j(m)$

2) $r \in S \Rightarrow (r,m) = 1$

3) $(a,m) = 1 \Rightarrow \exists r \in S \ni a \equiv r(m)$

If in addition, $0 \leqslant r \leqslant m$, S is a reduced set of least positive residues

Definition 6:- The Euler's ϕ-function is the number of positive integers not exceeding m, which are also coprime to m, i.e ,

$$\phi(m) = \sum_{\substack{(m,r)=1 \\ 0 < r \leq m}} 1$$

Eg , $\phi(1) = 1$, $\phi(2) = 1$, $\phi(4) = 2$, $\phi(11) = 10$

<u>Theorem</u> 4 - (Euler)

$$(a,m) = 1 \Rightarrow a^{\phi(m)} \equiv 1(m)$$

<u>Proof</u>:-

Let $r_1, r_2, \quad , r_k$ be a set of reduced residues modulo m

Then $a\,r_1, a\,r_2, \quad , a\,r_k$ also form a reduced set of residues if $(a,m) = 1$

Hence each of $r_1, r_2, \quad , r_k$ is congruent to some one of $ar_1, ar_2, \quad , ar_k$ So multiplying,

$$r_1 r_2 \quad r_k \equiv a^k r_1 r_2 \quad r_k (m)$$

But $(r_1 r_2 \quad r_k, m) = 1$ since $(r_i, m) = 1$ for all i So cancelling,

$$1 \equiv a^k (m)$$

But $k = \phi(m)$ by assumption Hence

$$a^{\phi(m)} \equiv 1(m)$$

Q E D

<u>Theorem</u> 5:- If $(a,m) = 1$, then $ax \equiv b(m)$ has a unique solution modulo m

Proof:- By Theorem 4,

$$(a,m) = 1 \Rightarrow a^{\phi(m)} \equiv 1(m)$$

Hence $a \; a^{\phi(m)-1} \equiv 1(m)$

So $a^{\phi(m)-1}$ is a solution of $ay \equiv 1(m)$ Multiplying both sides by b, and letting $x = by$,

$$a \; by \equiv b(m)$$

so that $x = by = ba^{\phi(m)-1}$ is a solution of $ax = b(m)$

To show that this solution is unique, let x_1 and x_2 be two solutions Then

$$ax_1 \equiv b, \quad ax_2 \equiv b \Rightarrow a(x_1 - x_2) \equiv 0(m) \Rightarrow m / a(x_1 - x_2)$$

Now since $(a,m) = 1$, $m \mid x_2 - x_1$ Hence $x_1 \equiv x_2(m)$

So the solution is unique modulo m

Q E D

Consider now

$$a_1x + b_1y \equiv c_1(m)$$
$$a_2x + b_2y \equiv c_2(m)$$

Theorems 1 and 2 enable Cramer's rule to be applied So, if

$$D = \begin{vmatrix} a_1 & b_1 \\ a_2 & b_2 \end{vmatrix}, \quad D_1 = \begin{vmatrix} c_1 & b_1 \\ c_2 & b_2 \end{vmatrix} \quad D_2 = \begin{vmatrix} a_1 & c_1 \\ a_2 & c_2 \end{vmatrix}$$

we have

$$Dx \equiv D_1(m)$$

$$Dy \equiv D_2(m)$$

If $(D, m) = 1$, then Theorem 5 is applicable, and

$$x = D_1 D^{\phi(m)-1} \qquad (m)$$

$$y = D_2 D^{\phi(m)-1} \qquad (m)$$

With the above background

$$\sum_1^{k+2} n_i w_i \equiv 0(11)$$

$$\sum_1^{k+2} n_i w_i' \equiv 0(11)$$

can be solved to get n_{k+1} and n_{k+2}

Rearranging,

$$w_{k+1} \quad n_{k+1} + w_{k+2} \quad n_{k+2} \equiv - \sum_1^k n_i w_i \ (11)$$

$$w_{k+1}' \ n_{k+1} + w_{k+2}' \quad n_{k+2} \equiv - \sum_1^k n_i w_i' (11)$$

Here

$$D = \begin{vmatrix} w_{k+1} & w_{k+2} \\ w_{k+1}' & w_{k+2}' \end{vmatrix}$$

$$D_1 = \begin{vmatrix} -\sum_1^k n_i w_i & w_{k+2} \\ -\sum_1^k n_i w_i' & w_{k+2}' \end{vmatrix}$$

$$D_2 = \begin{vmatrix} w_{k+1} & -\sum_1^k n_i w_i \\ w_{k+1}' & -\sum_1^k n_i w_i' \end{vmatrix}$$

So from

$$D\, n_{k+1} = D_1$$

$$D\, n_{k+2} = D_2$$

by Theorem 5,

$$n_{k+1} = D_1 \, D^{\phi(11)-1}$$

$$n_{k+2} = D_2 \, D^{\phi(11)-1}$$

A trick can be employed here to make $D = 1$, so that we have

$$n_{k+1} = D_1$$

$$n_{k+2} = D_2$$

For this it is only necessary to set $w_{k+1} = 2$, $w_{k+2} = 1$, $w_{k+1}' = 3$ and $w_{k+2}' = 2$, so that

$$D = \begin{vmatrix} 2 & 1 \\ 3 & 2 \end{vmatrix} = 1$$

So, finally,

$$n_{k+1} = D_1 = \left[\sim - \sum_{1}^{k} n_i w_i \quad + \sum_{1}^{k} n_i w_i' \right]_{11}$$

$$n_{k+2} = D_2 = \left[-2 \sum_{1}^{k} n_i w_i' \; + \; 3 \sum_{1}^{k} n_i w_i \right]_{11}$$

This seems to be the easiest method to find the check digits If the summations are to be found manually, the technique given in the last chapter can be used

3 4 <u>Discussion of the algorithm</u> :

This algorithm is completely different from all the error correcting codes It does not generate a syndrome which all the other algorithms do It requires two and only two check digits irrespective of the number of information digits, whereas in all the other codes, the number of check digits increases with the number of information digits It corrects two completely different types of errors whereas most of the codes can correct only one type of error

The reason why just two check digits are sufficient is that no syndrome is generated Hint is taken from the fact that there is only one way of correcting any given error The check digits are used as indicators to point out whether any alternation made is the

correction required or not

Of course it is possible to generate a syndrome to correct single errors (as G M Weinberg has done), and perhaps multiple errors also, but as experience has shown, they do not become popular in information systems due to the large overhead of check digits Another reason is that such conventional codes cannot correct exchange errors The proposed code corrects both these errors, which constitute perhaps 80 % of all the errors, with only two check digits The price to be paid for this is extra processing time on the computer The cost of this is probably commensurate with the benefits Also, since this error correction is to be done while reading, if we have a uni-programmed system or a multiprogrammed system with an imperfect program mix, then there will always be a lot of idle CPU time between the reading of two records, and the extra time required for error-correcting will be completely transparent to the user

CHAPTER 4

A COMPARISION WITH THE EXISTING DATA ENTRY METHODS

4 1 Overview -

In the last chapter, an error-correcting method was developed It remains to be seen how good a data entry method using the error-correcting feature will be, compared to the existing methods There are many factors which vary from method to method So if a quantitative index of performance is to be found then weightages or costs must be assigned to the factors Next, to quantify each factor, a model is needed for each of the methods The model should be quantitative, simple, and it should bring out all the essential differences in the methods

In this chapter the essential factors are enumerated, the index of performance is defined Then one model is developed for each of the methods, and expressions for the index are derived Finally the results are presented

4 2 Formulation of a model :-

The following methods are considered:-

1 Keypunch - error correcting

2 Keypunch - error detecting

3 Keypunch - verifying

4 Key-to-tape (and key -to-disk)

The main assumptions are

1 The overall error rate is independent of the number of records, or any other factor

2 Of the errors, 80 % are single transcription and single transposition errors

The essential parameters are:

N, the number of records

e, the error rate

L_i, the total number of characters in the important fields, for which strict error controls are justified

L_u, the total number of characters in unimportant fields

c, the number of important fields

L_c, the number of characters in the error control subfield, added to every important field

It can have values

0 - No error control

1 - Error detecting

2 - Error correcting

The length of a record is increased from $L_i + L_u$ to $L_i + L_u + c\,L_c$ due to the error control fields, and the number of characters to be entered, from $N(L_i + L_u)$ to $N(L_i + L_u + c\,L_c)$

The factors which should be derivable from the above parameters are

n, the number of "feedbacks", i e , number of times one has to go through the data entry process in order to enter a batch of N records

s, the extra keystrokes required

m, the extra manual handling in terms of characters

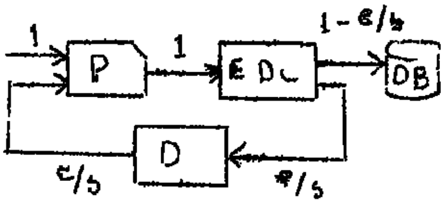
t, the extra time taken on the computer for error detecting and/or correcting

With these four factors, four costs c_n, c_s, c_m and c_t are associated, so that the expression for total cost is

$$C(i) = n\,c_n + s\,c_s + m\,c_m + t\,c_t$$

where i = 1,2,3,4 for the four methods

1 Keypunch-error correcting:-



P : Key punch

EDC Error detecting and correction

DB : Data base

D : Delay

N records are punched and fed to the computer Ne of them are erroneous Of them 80% , i e , 4Ne/5 are corrected, and Ne/5 are

sent back to be punched through the delay D. The same rule applies to the Ne/5 records, and so on, until one record is left. So the total number of records punched are

$$N' = N + \frac{Ne}{5} + \frac{Ne^2}{5^2} + \quad + \frac{Ne^n}{5^n}$$

where

$$\frac{Ne^n}{5^n} \leq 1 \quad \text{and} \quad \frac{Ne^{n-1}}{5^{n-1}} > 1$$

Solving, we get

$$n = \lceil (-\log N/\log(e/5))$$

Hence

$$N' = N(1-(e/5)^{n+1})/(1-(e/5))$$

The number of feedbacks is n. Each record actually contains $L_i + L_u + 2c$ characters, since two characters are to be added for each of the c important fields, that is, $L_c = 2$

The number of keystrokes required is

$$N(L_i + L_u + 2c)$$

But the actual number of keystrokes is

$$N(L_i + L_u + 2c)(1-(e/5)^{n+1})/(1- e/5)$$

So the extra keystrokes are

$$S = N(L_1 + L_u + 2c)((1-(e/5)^{n+1})/(1-(e/5)-1)$$

Since $L_c = 2$, the extra manual handling is $m = 2Nc$

The extra time taken on the computer is the time for the detecting algorithm to operate on all the records and the correcting algorithm on all the erroneous records The detection algorithm operates on

$$N' = N(1-(e/5)^{n+1})/(1-e/5)$$

records, and the correction algorithm on

$$Ne + \frac{Ne^2}{5} + \qquad + \frac{Ne^n}{5^n}$$

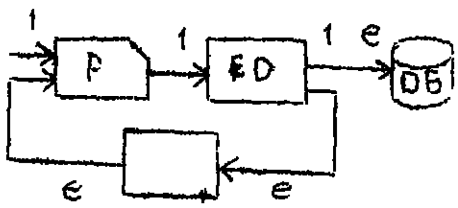
$$= Ne(1-(e/5)^n)/(1-e/5)$$

records So the extra time taken on the computer is

$$t = N(1-(e/5)^{n+1}) \text{ (time to detect error in c fields)}/(1-e/5)$$

$$+ Ne(1-(e/5)^n) \text{ (average time to correct one error)}/(1-e/5)$$

2 <u>Keypunch-error detecting:-</u>



P : Keypunch

ED Error-detecting

DB : Data base

D : Delay

Here the actual length of a record is $L_i + L_u + c$, since $L_c = 1$

Since the error rate is e, and no correction is involved, the total number of records actually punched is

$$N' = N + Ne + Ne^2 + \quad + Ne^n$$

where

$$Ne^n \leq 1 \quad \text{and} \quad Ne^{n-1} > 1,$$

so that the number of feedbacks

$$n = \lceil (-\log N/\log e)$$

Hence,

$$N' = N(1-e^{n+1})/(1-e)$$

The number of extra keystrokes is

$$S = N(L_i + L_u + c)((1-e^{n+1})/(1-e) - 1)$$

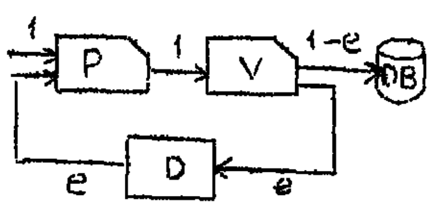
and the extra manual handling is

$$m = cN$$

characters The extra computer time is the time to detect error in the N' records actually fed, so that

$$t = N(1- e^{n+1})(\text{time to detect errors in c fields})/(1-e)$$

3 <u>Keypunch- verifying</u> -



P : Keypunch

V : Verifier

DB : Data base

D : Delay

Exactly as in keypunch-error detecting,

$$N' = N(1- e^{n+1})/(1-e)$$

where

$$n = \lceil (-\log N/\log e)$$

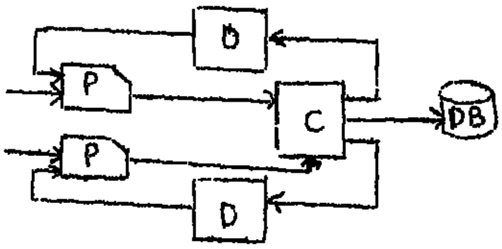
The number of extra keystrokes

$$S = N(L_1+ L_u) ((2(1-e^{n+1})/(1-e) - 1)$$

since $L_c = 0$

The extra manual handling and extra computer time are each zero

4 Key-to-tape -



K : Keyboard

C Comparator (a program)

DB Data base

D : Delay

It is assumed that the error rate e is same for both the operators, and that the errors do not overlap Also the difference in speeds of the two operators is not considered, since there is usually a buffer to take care of that

Each operator punches

$$N + 2Ne + \quad + 2Ne^{n}$$

records, so the total number of records punched are

$$N' = 2N + 4Ne + 4Ne^{2} + \quad + 4Ne^{n}$$

where

$$2Ne^{n} \leq 1 \qquad \text{and} \quad 2Ne^{n-1} > 1$$

The number n is

$$n = \lceil (-\log 2N/\log e)$$

The total number of keystrokes are

$$N(L_I + L_u)(2 + 4e + 4e^{2} + \quad + 4e^{n})$$

$= N(L_l + L_u)(4(1-e^{n+1})/(1-e)-2)$

Since $N(L_l+L_u)$ strokes were actually required, the extra keystrokes are

$$S = N(L_l+L_u)(4(1-e^{n+1})/(1-e)-3)$$

The extra manual handling $m = 0$ The extra computer time the time to compare

$$N + 2Ne + \qquad + 2Ne^{n}$$

pairs of records So

$t = N(2(1-e^{n+1})/(1-e)-1)$ (time to compare two records)

Now the expressions for cost

$$C(i) = n\ c_n + s\ c_s + m\ c_m + t\ c_t$$

can be written down as

$$C(1) = n\ c_n + N(L_l+L_u + 2c)((1-(e/5)^{n+1})/(1- e/5)-1)\ c_s + 2Nc\ c_m$$
$$+ (N(1-(e/5)^{n+1}\ \text{(time to detect error in c fields)}/(1- e/5)$$
$$+ Ne(1- (e/5)^{n+1})\ \text{(average time to correct one error)}/(1- e/5))\ c_t$$

$$C(2) = n\ c_n + N(L_l+ L_u +c)((1- e^{n+1})/(1-e)-1)\ c_s + Nc\ c_m$$
$$+ N(1- e^{n+1})\text{(time to detect errors in c fields)}/(1-e)\ c_t$$

$C(3) = n\ c_n + N(L_l + L_u)(2(1-e^{n+1})/(1-e) - 1)\ c_s$

$C(4) = n\ c_n + N(L_l + L_u)(4(1-e^{n+1})/(1-e) - 3)\ c_s + N(2(1-e^{n+1})/(1-e) - 1)$

(time to compare 2 records) c_t

For 1,

$n = \lceil (-\log N/\log(e/5))$

For 2 and 3,

$n = \lceil (-\log N/\log e)$

For 4

$n = \lceil (-\log 2N/\log e)$

Now in most of the applications, the extra manual handling does not cost much, and the extra computer time is not important since it is very small So it may be assumed for simplicity that $c_c = c_t = 0$ To effect another simplification, let the percentage of extra keystrokes be considered, instead of the actual number of extra keystrokes Also, let $c_n = 1\ 0$ and $c_s = 0\ 01$, implying that one extra feedback costs 100 times more than one percent extra keystrokes In fact the ratio c_n/c_s could be much more Then

$C(1) = n + (1 + 2c/(L_l + L_u))((1 - (e/5)^{n+1})/(1 - e/5) - 1)$

$C(2) = n + (1 + c/(L_l + L_u))((1 - e^{n+1})/(1-e) - 1)$

$C(3) = n + 2(1-e^{n+1})/(1-e) - 1$

$C(4) = n + 4(1- e^{n+1})/(1-e) - 3$

with the n in each case unchanged

Example -

Let

No of records, $N = 10{,}000$

Error rate , $e = 1\,\%$

Length of important fields, $L_i = 20$

Length of unimportant fields, $L_u = 80$

No of important fields, $c = 2$

Then applying the above formulas, roughly

$C(1) = 2$ units

$C(2) = 3$ units

$C(3) = 4$ units

$C(4) = 4$ units

4 3 Results -

The graphs appended show the results $L_i + L_u$ has been taken to be 100 throughout c is taken to be 1 and 2, and for each, e is taken to be 1 %, 2%, , 10% and graphs are plotted of cost vs N, which varies from 1000 to 10,000 in steps of 1000

The conclusions from the graphs are

1 The method with the error-correcting code costs less than each of the others

2 As the error rate goes up, the superiority of the proposed method becomes more marked

So it can be concluded, that with reasonable assumptions, incorporation of the proposed error correcting code in a data entry method will decrease the overall cost

CHAPTER 5

SECRECY TRANSFORMATIONS

5 1 Overview:-

This chapter deals with the problem of secrecy transformations in information systems Many files, especially in integrated systems, contain sensitive data which must be protected The problem becomes more severe in shared data bases In many applications, secret data has to be handled manually, and it becomes necessary to code the data

The question of restricting access to files has received some attention, but no simple and efficient means have been developed to code data in order to render it unintelligible This chapter lists the requirements of secrecy transformations, presents several simple algorithms for the purpose Finally it give simple quantitative methods of comparing the transformations in view of the requirements

The concepts developed were applied to a live system, described in chapter 6

5 2 Some proposed transformations -

The main requirements can be stated as below -

1) The code should be invertible

2) The coded number should bear as little relation to the original number as possible

3) The algorithm should be easily implementable

4) The transformation should preserve all the error detecting and correcting properties of the original number

5) The code should be as difficult to break as possible

6) The algorithm should suit the number representation of the original number

7) The transformation should have parameters which can be easily altered without changing the algorithm

Which of the following algorithms should be chosen depends on the cost-effectiveness of the particular situation For example a code suitable for military information transmission may not be suited for postal transmissions, even though the same type of information is transmitted in both the cases

In the following algorithms, only numeric data of fixed length has been considered, but extensions are easy Transformations T are presented,

$$\langle a_1 \quad a_2 \qquad a_n \rangle \xrightarrow{T} \langle b_1 \; b_2 \qquad b_m \rangle$$

where $a_i \in \{0, 1, 2, \quad , 9\}$ for $i = 1, 2, \quad , n$ and the b_i s could be digits, alphabets or other symbols

Algorithm 1 Permutation :-

Here T is an identity matrix of order n with its columns permuted, so that

$$\langle a_1 \; a_2 \quad a_n \rangle \xrightarrow{T} \langle b_1 \; b_2 \quad b_n \rangle$$

with b_i = some a_j uniquely

This algorithm does not require matrix multiplication, and is in fact, very simple, The parameter is the permutation sequence, having n! different values All the error control features of the original number are fully preserved If the BCD representation is used, a FORTRAN program for this will require only six statements

Algorithm 2 : Change of base :-

Here the base of the number is changed from 10 to some greater or smaller integer If a base greater than 10 is chosen, then instead of using A,B,C, for 10,11,12, , alphabets and other symbols could be chosen at random, thus making code breaking more difficult This transformation, however, may not preserve the error detecting and correcting properties

Algorithm 3: 9's complement:-

In this algorithm, the digitwise 9's complement of the number is taken This algorithm is extremely simple both for the BCD

and binary representations It preserves the error detecting and correcting properties But this transformation has no parameters

Algorithm 4 Generalized complementation -

In this algorithm, the number is subtracted from a constant larger than all the numbers in the batch (Alternatively, a constant could be added to or subtracted from the number)

This has all the advantages of Algorithm 3, and in addition, it has a parameter which can be easily varied

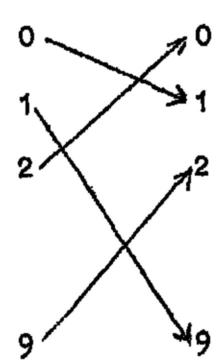
Algorithm 5 Assigning digits:-

a) Without table lookup :-

Here every digit a_i is replaced by $[a_i + k]_{10}$ where k is any integer from 1 to 9

b) With table lookup :-

Here a mapping is stored in the form of a table, like



10 ! such mappings are possible, and it is very easy to go from one to another

Both these preserve the error control properties

Algorithm 6: Assigning several characters:-

Here we have a one-to-many mapping, e g

$1 \rightarrow$ A Y Z 2

$2 \rightarrow$ P R D 9 , etc

To transform any digit, one out of these 4 characters is chosen at random This serves to make code breaking extremely difficult As a special case we could have only one character instead of four,

Algorithm 7 Random number generation:-

This requires a good random number generator which does not give repeated values For each number to be transformed in a batch, a random number is generated, and a table of the original number and the random number is maintained This is necessary because the algorithm is not invertible Its main advantage is that there is absolutely no way to break the code, for the simple reason that there is no code It requires an appreciable time on the computer

Algorithm 8: A number dependent coding:-

If the number has n digits, choose an integer k, $1 < k \leq n$ Then the algorithm is

$$a_i \longrightarrow \{a_i + a_k\}_{10}\,, \qquad i = 1,2, \quad ,n$$
$$i \neq k$$
$$a_k \longrightarrow a_k$$

Here, effectively, the algorithm changes from number to number In addition, k can be varied It preserves the error control properties

Algorithm 9: Number dependent rotation -

Find

$$k = \left\lceil \sum_{1}^{n} a_i \right\rceil_n$$

and rotate the number right or left by K places

Here also, effectively, the algorithm is number dependent In general, any symmetric function $f(a_1, a_2, \quad , a_n)$ which gives integer values can be used

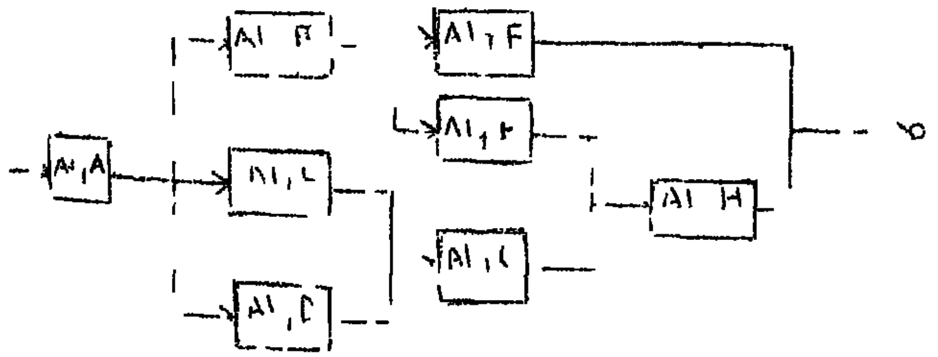
Algorithm 10: Several algorithms in series:-

Since each transformation is invertible, any number of algorithms could be applied one after the other

Algorithm 11: Several algorithms in parallel:-

The n digit number could be divided arbitrarily into subfields, and different algorithms could be applied to the different fields The subfields need not contain consequitive digits

In the most sophisticated case, we could have a series-parallel structure of algorithms, e g , as shown below



5 3 <u>Evaluation of transformations:-</u>

Since there are many requirements of secrecy transformations, and specific algorithms satisfy them to a varying degree, a quantitative measure is required to properly evaluate them

One obvious way is to define an index

$$I = c_1f_1 + c_2f_2 + \quad + c_nf_n$$

where $f_1, f_2, \; , f_n$ are the factors involved, and $c_1, c_2, , \; , c_n$ are the costs associated The factors could be time taken, storage required, retention of error control properties, etc But it is not feasible to employ this method practically, since all the factors cannot be easily quantified

However, given any two algorithms, it is relatively easy to say which factor is more favourable in one of them compared to

the other Based on this fact, a simple scheme is devised

First, n factors are identified The set of algorithms $A_1, A_2, \ldots, A_m$ is considered For any two algorithms A_i and A_j, an n-dimensional comparision vector C is defined

$$C = \left[c_1\ c_2\ \ldots\ c_n\right]$$

Here $c_k = 1$ if the kth factor of A_i is 'better' than the kth factor of A_j Otherwise it is zero Now A_i is better than A_j, if C for A_i and A_j contains more 1's than 0's

With the above comparision the set of algorithms can be sorted to give the sequence $A^1, A^2, \ldots, A^n$, which are in decreasing order of "goodness'

If a weightage could be given to every factor, then that could be used instead of 1 and 0

CHAPTER 6

A CASE STUDY

6 1 Overview:-

The aim of this case study is to demonstrate the usefulness of the data error controls and secrecy transformations described in the previous chapters The case chosen was the data processing of the Joint Entrance Examinations for entrance to the five IIT's and B H U It was chosen because it highlights the two main areas which form the subject matter of this thesis

After a description of the system, the two aspects will be dealt with Detailed description of the other aspects is omitted because they are not of direct interest here

6 2 Description of the system:-

Every year, one of the IITs is put in charge of the data processing activity That IIT takes the responsibility, develops the programs (This is done every year !) processes all the data, and produces the final admission lists In the year 1974-75, IIT Kanpur was put in charge A generalized system was developed which can be used in the subsequent years, irrespective of the IIT in charge and the computer used

The steps in the system are outlined below (the dates/months are omitted)

1 Advertisements appear in the major newspapers The applicants request for application forms, get them, fill them, and send them to the I I T in whose zone they wish to appear for the examination, irrespective of the IIT which they wish to join

2 At each of the IITs, applications are sorted manually first examination centerwise, and then groupwise

3 Registration numbers are allotted as follows

IIT Bombay	:	10001	to	19999
IIT Delhi	:	20001	to	29999
IIT Kanpur	:	30001	to	39999
IIT Kharagpur	:	40001	to	49999
IIT Madras	:	50001	to	59999

Each IIT allots registration numbers to its centers in groups of 100 Sufficient gap is left out between the last registration number of the group A candidates and the first registration number of the group B candidates for the same center, and also between two centers This is to make additions possible

Example :-

For Ahmedabad center, suppose there are 107 candidates from group and 34 from group B Then the center gets the slice 1001 to

10200 Group A candidates are given numbers 10001 to 10107 and group B, 10151 to 10184

The next center, Rajkot, gets numbers starting from 10201

4 Three copies of the center-wise and group-wise numbering schemes evolved at the IITs, as given in (3) above, are sent to the IIT in charge

5 At each IIT, the basic information sheet is seperated from every application form, completed if necessary and possible, scrutinized, and cards are punched from it The scrutinizing and punching can go on simultaneously Each IIT has to buy its own requirements of cards

6 The cards are packed in seperate bundles for each center and group, sorted in the ascending order of registration numbers Centerwise and groupwise listings of the cards are made in duplicate, and a final scrutiny is made

7 The cards, arranged as in (6), together with one copy of the listing is sent to the IIT in charge, along with the three copies of the numbering scheme, as in (4), through an authorized representative

8 The computer center at the IIT in charge loads the cards onto a tape The errors discovered are corrected in consultation with the representatives of the IITs Finally a verification list

is produced which is certified by the representatives Four copies of roll lists are prepared Each contains the applicants registration number, name, and space for him or her to sign at the time of each examination Three copies are sent to the IITs through the representatives One is retained at the IITs, one is sent by post to the presiding officer of each examination center, and the last one is taken to the center by a representative

9 Next, coding-cum-tabulation sheets are prepared There are printed on multi-part continuous pre-printed stationery The first part contains the applicant s registration number, name and the secret code generated by the computer Then there is a part for each subject containing the code and space for writing the marks At the bottom of each page, there is space for writing the total of marks (for each subject) These forms are printed, sealed, and sent to the Chairman of the admissions committee of each IIT, through responsible persons They are opened only at the time of transcribing the codes onto the answer scripts,

10 After the examinations, the answer scripts, along with the roll lists bearing the signatures of the candidates who were present, are brought to the respective IITs, through responsible persons The code-cum-tabulation sheets are opened, and the

code transcribers enter the codes at two places on each answer book One of them contains the applicant's name and registration number which is torn off, and the other is retained on the answer script which is sent to the examiners The part torn off, together with the first part of the code-cum-tabulation sheets, is stored in a secure place by the Chairman Before transcribing it is ensured that the name on the answer scripts and the tabulation sheet, and also the two registr tion numbers, are identical All this is done for answer scripts of each subject, and they are then sorted in the order in which they appear in the tabulation sheets

11 The answer scripts, in the order mentioned in (10), together with the part of the tabulation sheet for that subject are -sent to the head examiner He distributes them to the examiners who, after evaluation, enter marks against the code numbers They also total the marks for every page Then the scrutinizers look up the marks on the answer sheets, and enter them against the code numbers of a second copy of the code-cum-tabulation sheets, independently The two lists are compared, and the errors are corrected The answer books, together with two sets of marksheets signed by the examiners and scrutinizers are sent to the Chairman

12 The parts for each subject bearing the same page number are stuck together with cello tape and the form is reconstructed The two copies are tallied and the information is transcribed to the third copy The copy is sent to the IIT in charge

13 Cards are punched at the IIT in charge The computer prepares merit lists seperately for SC, ST and other candidates, irrespective of whether the applicants belong to group A or B Also zonal merit lists are produced Cards are punched only for those SC/ST candidates who have appeared for all the examinations and the other candidates who have secured at least 25 % in English and 30 % in the other subjects For preparing the merit lists, English marks are not considered In case of a tie, marks in Mathematics are considered If still the tie persists, marks in Physics for group A and in Physics and Chemistry for group B candidates are considered If the tie still persists, same ranking is given, but one is added to the next rank

Eg - If two candidates are tied up and one gets the rank 174, then the other will also get 174, but the next candidate will get the rank 176 , not 175

14 A meeting of the chairman of the admissions committee decide the cut off point for calling candidates for interview They take the lists and go back to the IITs where zonal merit lists are prepared manually The two lists are tallied and the errors corrected

15 Interview letters are sent to the candidates, and after the interviews, final selections are made

For last minute changes in roll lists on account of changes in center or group, or addition of new candidates, new registration numbers are assigned, and codes are assigned from a coding table sent to the chairman of the admissions committees

Some parts of the system, such as getting the question papers, are secret, and cannot be revealed Anyway, they are out of the purview of the computer data processing

<u>Secrecy transformations</u> -

The registration number is numeric and has 5 digits The first digit can take 5 values representing the 5 IITs, as follows:

1 IIT Bombay

2 IIT Delhi

3 IIT Kanpur and B H U

4 IIT Kharagpur

5 IIT Madras

The other four digits are not continuous, since for every center, the number begins with the next hundreds place (in order to leave gaps for emergency registrations, transfers, etc)

It was required to transform this to a 4-character alphanumeric code (first two alpha, and the last two numeric) such

that the identity of the IITs was preserved The codes were to be generated by the computer The following algorithm was chosen

The first digit was transformed as

1 ⟶ A, B, C, or D

2 ⟶ E, F, G, or H

3 ⟶ J, K, L, or M

4 ⟶ N, P, Q or R

5 ⟶ S, T, U or V

which of these 4 characters are to be used is decided by the next two digits

Twenty five different alphabets in a random order are chosen (QUICK) The second and third digits can have values 00 to 99 Twenty five of these hundred are associated with one of the four digits

<u>Eg</u> - For IIT Bombay,

101xx to 125xx ⟶ Aaxx

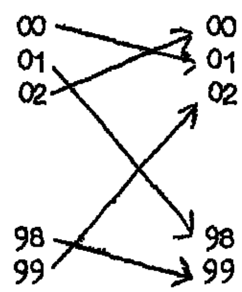
126xx to 150xx ⟶ Baxx

151xx to 175xx ⟶ Caxx

170xx to 200xx ⟶ Daxx

where a is one of the 25 alphabets QUICK , and xx represent the last two digits

Coding for the last two digits is one by a one-one into mapping



Eg - The registration number 10199 becomes AQ02, since the mappings are

$$101 \rightarrow AQ$$
$$11 \rightarrow 02$$

A one-page FORTRAN subroutine performs the transformation both ways

Error control -

1 Almost every manual task is done twice and the results are tallied The errors are corrected manually

2 In the roll-cum-tabulation sheets, after the marks are entered and they are stuck together by cello tape, the marks in every row are added The marks in every column are already added by the examiners The grand total of both these types of totals should tally, for every page

3 The registration numbers are assigned sequentially When they are brought to the IIT in charge they are fed to an edit program The program reads the cards and discovers the missing and duplicate cards Some of the errors can be corrected at once For example,

if the group is punched wrongly, say B is punched for A, then this can be found from the range in which the registration number lies

Or, if one registration number does not confirm to the sequence, e g ,

, 10123, 10124, 10124, 10525, 10126,

then the 5 can be corrected and put as '1'

But inspite of all these checks and corrections, it has been observed that there are some errors which cannot be corrected Then the basic information sheets of the application forms have to be consulted Since altogether there are about 20,000 applicants every year, those many sheets cannot be brought to the IIT in charge Moreover, rules do not permit moving them So in order to correct these errors, the representatives have to go back to their IITs And since the examinations have to be held on schedule, this creates problems

So it is not a question of just detecting the errors, they have to be corrected The error-correcting scheme is required only for registration numbers Errors in the name field can be tolerated and errors in the group field can be corrected This leaves only the field specifying SC/ST Since their number is small this can be checked manually, even after the examinations

The registration number will have to be increased by two digits The scheme described in the earlier chapters can be directly used The weights are as below:

Digit	7	6	5	4	3	2	1
W	9	3	7	8	5	2	1
W'	8	9	6	1	7	3	2

The coding, error detecting and correcting algorithms are exactly the same

In conclusion, it may be said that it is imperative to correct the errors in time, because even one error in the 20,000 records may mean ruining the career of a good student

CHAPTER 7

CONCLUSION

Though information systems have proliferated widely in recent years, and topics like system analysis and design, management of projects, have received wide attention, the control aspect has been neglected, and unjustifiably so In the present work, two aspects of this problem, namely data error control and secrecy transformations, have been looked into

It was found that input data editing procedures were naive and inadequate They tend to increase the overall time for entering a batch of data, and necessitate many more keystrokes than actually required, especially if the error rate is high It was also observed that there does not exist any error correcting code which can be profitably incorporated into information systems to solve this problem A simple and elegent error correcting code was invented which was suitable for the purpose With the help of a model it was shown that data entry methods having this feature will do better than the existing methods, under reasonable assumptions

In the area of secrecy transformations, the need was for simple and efficient codes which at the same time should be difficult to break Several basic algorithms were presented, and it was

indicated how codes of any desired degree of sophistication can be built from them by series-parallel combinations In order to choose objectively from amongst the codes, a simple technique was evolved to compare the code, taking note of the fact that many of the factors involved are subjective

After a brief survey of the various application areas, it is felt that for any file, the key field can economically have the error -correcting code Also, for such other important fields like cash, inventory level, etc , introduction of the error-correcting code could turn out to be profitable, depending upon the actual costs In some areas like defence information systems, medical information systems, spacecraft control where errors cannot be tolerated and turn-around time is critical, the error correcting code could be very useful

The secrecy transformations developed can find ready application in defence intelligence, examination data processing, shared direct access files used by competing business concerns, company files containing personnel evaluation, bank information systems containing the credit-worthiness of customers, large national data banks, etc

B I B L I O G R A P H Y

1 Andrews, A M , A varient of modulus 11 checking, The Comp Bull , Aug 70, pp 261-5

2 Baran, P , On distributed communications IX security, secrecy and tamper-free considerations, Doc RM-3765-PR, Rand Corp Aug 64

3 Beardsley, C W , Is your computer insecure ?, IEEE Spectrum, Vol 9, No 1(Jan 72), pp 67-8

4 Beckley, D F , An optimum system with "modulus 11", The Comp Bull , Dec 67, pp 213-5

5 Beckley, D F , check digit verification, Data Processing, Jul -Aug 1966, pp 194-201

6 Bingham, H W , Security techniques for EDP of multilevel classified information, Doc RADC-TR-65-415, Rome Air Dev Centre, Dec 65

7 Boyd, R , Keyboard and keyboard problems, Computer Typesetting Conf Proc , Jul 64, pp 152-73

8 Briggs, T , Modulus 11 check digit systems, The Comp Bull , Aug 70, pp 266-9

9 Bulyk, C H , You cant have too many controls, Systems, No 1, pp 20-1

10 Burton, R W , et -al , Use of edit information to measure and reduce file error content in a MIS, Decision Studies Group, AD-736210, Oct 71, 60p

11 Campbell, D V A , A modulus 11 check digit system for a given system of codes, The Comp Bull , Jan 70, pp 12-3

12 Campbell, D V A , Check digits, The Comp Bull , Mar 70,p 71

13 Carlson, G , Predicting clerical error, Datamation, Vol 9,No 2 (Feb 63), pp 109-11

14 Carrol, J M , et-al , Fast infinite key privacy transformation for resource sharing systems Proc AFIPS 1970,FJCC,Vol 36, pp223-30

15 Carrol, J M , et al , Multidimensional security program for a generalized information retrieval system, Proc AFIPS 1971, FJCC,pp 571-77

16 Conway, R W , et-al On the implementation of security measures in information systems, C-ACM,Vol 15,No 4(Apr 72),pp 211-20

17 Conway, R W , et-al Selective security capabilities in ASAP- a file management system, Proc AFIPS 1972, SJCC

18 d'Agapeyeff, A , Computing, privacy and the public welfare, The Comp Bull Suppl , Vol 13,No 12(Dec 70),pp vi-vii

19 Dimsdale, B , et-al , Programmed error correction in project mercury, C-ACM, Vol 3, No 12 (Dec 60),pp 649-52

20 Friedman, T D , The authorization in shared files, IBM Sys Jl , Vol 9, No 4 (Apr 70), pp 258-80

21 Garrision, W A , et-al , Privacy and security in data banks, Taxas Univ AD-718406, Nov 70

22 Graham, R.M , Protection in an information processing utility, C-ACM, Vol 11,No 5(May 68),pp 365-9

34 Peterson, H E , et-al , System implications of information privacy, Proc AFIPS 1967, SJCC, Vol 30, pp291-300

35 Rathery, B , The collection of information, Data Processing Magazine, Nov 66, pp 62-3

36 Reid, C J , Modulus 11 check digits, The Comp Bull., Apr 70, p 122

37 Scaletta, P J ,The computer as a threat to individual privacy, Data Management, Vol 9, No 1 (Jan 71), pp 18-23

38 Shirley, D , Data collection, Data Processing Magazine, Sept 66, pp 20-25

39 Skatrud, R O , The application of cryptographic techniques to data processing, Proc AFIPS 1969, FJCC,Vol 34,pp 111-7

40 Smith, N J , Automated information preparation, Jl of Comm , Vol 12, No 6,(Jun 62), pp 84-9

41 Sullivan, A , Controls for a complex business system, Data Processing Magazine, Vol 7, No 12 (Dec 65) pp 22-30

42 Teeple, J B , Reducing human error in ADP systems, Datamation, Vol 6,No 6 (Nov-Dec 60), pp 91-4

43 Turn, R , et-al , Security of a computerized information system, Rand Corp AD-709366, July 70, 9p

44 Weinberg, G M , Programmed error correction on a decimal computer, C-ACM, Vol 4, No 2,(Feb 61), pp 174-5

45 Westin, A , Privacy and freedom, Atheneum, New York, 1967

46 Wild, W G , The theory of modulus 11 check digit systems, The Comp Bull , Dec 68, pp 309-11

47 Wilson, J G , Notes on geometric weighted check digit verification, C-ACM, Vol 4, No 5 (May 61), pp 551-2

48 ________, Security identification system- specification and guides, Tech Bull , Committee on Uniform Security and Identification Procedures, Dept of Automation, American Bankers Assoc

49 ______I___ , CODASYL data base task force report, rev , Apr 71 (available ACM headquarters)

****